Troubleshoot EtherChannels on Catalyst 9000 Switches

Contents

Introduction

This document describes how to understand and troubleshoot EtherChannels on Catalyst 9000 series switches.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Catalyst 9000 Series Switches Architecture
- Cisco IOS® XE Software Architecture
- Link Aggregation Control Protocol (LACP) and Port Aggregation Protocol (PAgP)

Components Used

The information in this document is based on these hardware versions:

- Catalyst 9200

- Catalyst 9300
- Catalyst 9400
- Catalyst 9500
- Catalyst 9600

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

Please refer to the Cisco Official Release Notes and Configuration Guides for up-to-date information about the limitations, restrictions, configuration options, and caveats as well as any other relevant details about this feature.

EtherChannel provides fault-tolerant high-speed links between switches, routers, and servers. Use the EtherChannel to increase the bandwidth between devices, and deploy it anywhere in the network where bottlenecks are likely to occur. EtherChannel provides automatic recovery for the loss of a link, it redistributes the load across the remaining links. If a link fails, EtherChannel redirects traffic from the failed link to the remaining links in the channel without intervention.

EtherChannels can be configured with no negotiation or dynamically negotiate with the support of a Link Aggregation Protocol, either PAgP or LACP.

When you enable PAgP or LACP, a switch learns the identity of partners and the capabilities of each interface. The switch then dynamically groups interfaces with similar configurations into a single logical link (channel or aggregate port); the switch bases these interface groups on hardware, administrative, and port parameter constraints.

LACP Flags

LACP flags are used to negotiate port-channel parameters when it comes up. Have a look at the meaning of every flag:

Flag	**Status**
LACP Activity (less significant bit)	0 = Passive mode 1 = Active mode
LACP Timeout: Indicates the LACP sent/received timeout	0 = Long timeout. 3 x 30 sec (default) 1 = Short timeout. 3 x 1 sec (LACP rate fast)
Aggregation	0 = Individual link (not considered for aggregation) 1 = Aggregatable (potential candidate for aggregation)

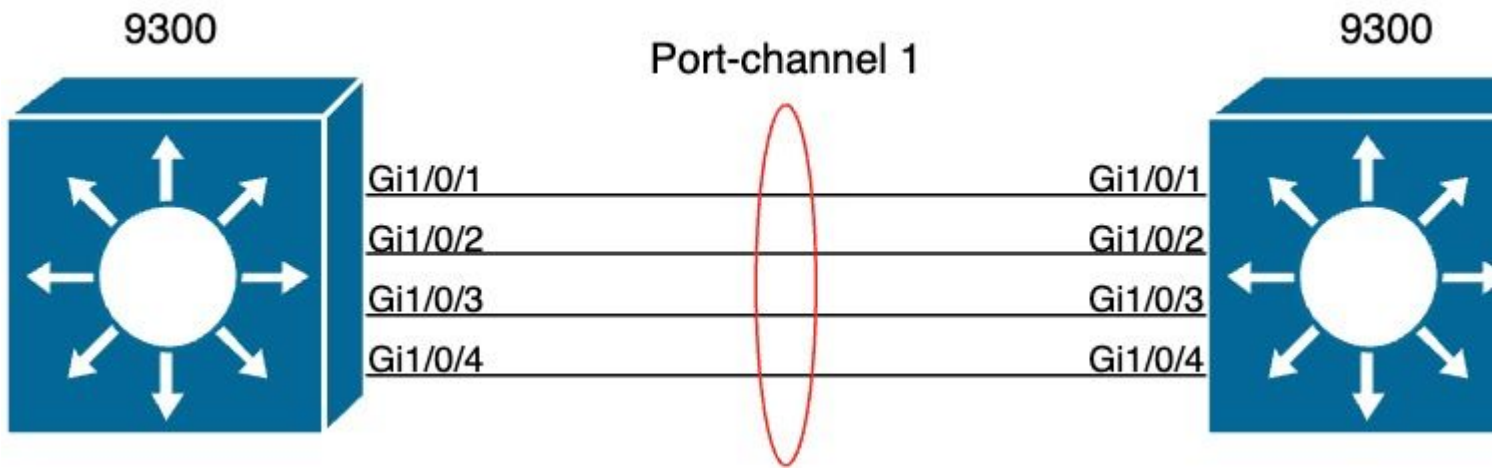
Synchronization	0 = The link is out of Sync (non-good state) 1 = The link is in Sync (good state)
Collecting	0 = Not ready to receive/process the frames 1 = Ready to receive/process the frames
Distributing	0 = Not ready to send/transmit the frames 1 = Ready to send/transmit the frames
Defaulted	0 = It uses the information in the received PDU for the partner 1 = It uses default info for Partner
Expired (most significant bit)	0 = PDU is expired, 1 = PDU is valid

The expected value for LACP flags is 0x3D (hex) or 0111101 (binary) to reach the P (bundled in port-channel) status.

```
.... ...1 = LACP Activity (less significant bit)
.... ..0. = LACP Timeout
.... .1.. = Aggregation
.... 1... = Synchronization

...1 .... = Collecting
..1. .... = Distributing
.0.. .... = Defaulted
0... .... = Expired (most significant bit)
```

Network Diagram



Verify LACP Operation

This section describes how to verify the correct state and operation of the LACP protocol.

Basic Checks

Check the LACP outputs with these commands:

<#root>

```
show lacp sys-id

show lacp <channel-group number> neighbor

show lacp <channel-group number> counters

show interfaces <interface ID> accounting

debug lacp [event|packet|fsm|misc]

debug condition <condition>
```

The first command output displays the switch system ID and its priority (for LACP).

<#root>

```
switch#
show lacp sys-id

32768,
```

```
f04a.0206.1900 <-- Your system MAC address
```

Check the details of the LACP neighbor, such as the operational mode, neighbor system Dev ID, and its priority.

<#root>

```
switch#
show lacp 1 neighbor


Flags:  S - Device is requesting Slow LACPDUs
        F - Device is requesting Fast LACPDUs
        A - Device is in Active mode       P - Device is in Passive mode

Channel group 1 neighbors

                     LACP port                      Admin  Oper   Port    Port
Port          Flags  Priority
Dev ID

          Age  key    Key    Number  State
Gi1/0/1       SA     32768
f04a.0205.d600

  12s  0x0    0x1    0x102   0x3D
<-- Dev ID: Neighbor MAC Address

Gi1/0/2       SA     32768
f04a.0205.d600

  24s  0x0    0x1    0x103   0x3D
<-- Dev ID: Neighbor MAC Address


Gi1/0/3       SA     32768
f04a.0205.d600

  16s  0x0    0x1    0x104   0x3D
<-- Dev ID: Neighbor MAC Address

Gi1/0/4       SA     32768
f04a.0205.d600

  24s  0x0    0x1    0x105   0x3D
<-- Dev ID: Neighbor MAC Address
```

Validate LACP packets sent and received by each interface. If corrupt LACP packets are detected, the Pkts Err counter increases.

<#root>

```
switch#
show lacp 1 counters

                 LACPDUs         Marker          Marker Response  LACPDUs
Port             Sent   Recv     Sent   Recv     Sent   Recv      Pkts Err
---------------------------------------------------------------------------
Channel group: 1
Gi1/0/1
3111   3085
     0      0        0      0
0

Gi1/0/2
3075   3057
     0      0        0      0
0

Gi1/0/3
3081   3060
     0      0        0      0
0

Gi1/0/4
3076   3046
     0      0        0      0
0
```

There is also an option to check the interface accounting for LACP.

<#root>

```
switch#
show int gi1/0/1 accounting

GigabitEthernet1/0/1
                Protocol    Pkts In   Chars In   Pkts Out  Chars Out
                   Other          0          0      10677     640620
                    PAgP        879      78231        891      79299
           Spanning Tree        240      12720         85       5100
                     CDP       2179     936495       2180     937020
                     DTP       3545     170160       3545     212700
                    LACP       3102     384648       3127     387748
```

Debugs

When there is no LACP sync-up or when the remote peer does not run LACP, Syslog messages are generated.

```
%ETC-5-L3DONTBNDL2: Gig1/0/1 suspended: LACP currently not enabled on the remote port.
%ETC-5-L3DONTBNDL2: Gig/1/0/1 suspended: LACP currently not enabled on the remote port.
```

Enable LACP debugs with the use of these commands:

<#root>

```
debug lacp [event|packet|fsm|misc]
```

```
debug condition <condition>
```

If you notice LACP negotiation issues, **enable** LACP debugs to analyze why.

<#root>

```
switch#
debug lacp event

Link Aggregation Control Protocol events debugging is on
switch#
debug lacp packet

Link Aggregation Control Protocol packet debugging is on
switch#
debug lacp fsm

Link Aggregation Control Protocol fsm debugging is on
switch#
debug lacp misc

Link Aggregation Control Protocol miscellaneous debugging is on
```

If needed, also **enable** debug condition to a specific interface and filter the output.

<#root>

```
switch#
debug condition interface gigabitEthernet 1/0/1
```

Note: LACP debugs are platform agnostic.

Validate debugs and filters are set up.

<#root>

```
switch#
show debugging

Packet Infra debugs:

Ip Address                                                Port
-------------------------------------------------------|----------

LACP:
  Link Aggregation Control Protocol
miscellaneous
 debugging is
on
  Link Aggregation Control Protocol
packet
 debugging is
on
  Link Aggregation Control Protocol
fsm
 debugging is
on
  Link Aggregation Control Protocol
events
 debugging is
on

Condition 1: interface Gi1/0/1 (1 flags triggered)
        Flags: Gi1/0/1
```

Analyze the LACP debugs, and use the **show logging** command to display them. The debug output shows the last LACP frames before the port-channel interface comes up:

<#root>

```
switch#

show logging

<omitted output>

LACP :lacp_bugpak: Send LACP-PDU packet via Gi1/0/1
LACP : packet size: 124

LACP: pdu: subtype: 1, version: 1

LACP: Act: tlv:1, tlv-len:20, key:0x1, p-pri:0x8000, p:0x102, p-state:0x3D, s-pri:0x8000, s-mac:f04a.020
LACP: Part: tlv:2, tlv-len:20, key:0x1, p-pri:0x8000, p:0x102, p-state:0xF, s-pri:0x8000, s-mac:f04a.020

LACP: col-tlv:3, col-tlv-len:16, col-max-d:0x8000
LACP: term-tlv:0 termr-tlv-len:0
LACP: HA: Attempt to sync events -- no action  (event type 0x1)


LACP :lacp_bugpak: Receive LACP-PDU packet via Gi1/0/1
LACP : packet size: 124

LACP: pdu: subtype: 1, version: 1

LACP: Act: tlv:1, tlv-len:20, key:0x1, p-pri:0x8000, p:0x102, p-state:0x3D, s-pri:0x8000, s-mac:f04a.020
LACP: Part: tlv:2, tlv-len:20, key:0x1, p-pri:0x8000, p:0x102, p-state:0x3D, s-pri:0x8000, s-mac:f04a.02

LACP: col-tlv:3, col-tlv-len:16, col-max-d:0x8000
LACP: term-tlv:0 termr-tlv-len:0

LACP: Gi1/0/1 LACP packet received, processing    <-- beginning to process LACP PDU

    lacp_rx Gi1/0/1 - rx: during state CURRENT, got event 5(recv_lacpdu)
@@@ lacp_rx Gi1/0/1 - rx: CURRENT -> CURRENT
LACP: Gi1/0/1 lacp_action_rx_current entered

LACP: recordPDU Gi1/0/1 LACP PDU Rcvd. Partners oper state is hex F    <-- operational state

LACP: Gi1/0/1 partner timeout mode changed to 0
    lacp_ptx Gi1/0/1 - ptx: during state FAST_PERIODIC, got event 2(long_timeout)
@@@ lacp_ptx Gi1/0/1 - ptx: FAST_PERIODIC -> SLOW_PERIODIC
LACP: Gi1/0/1 lacp_action_ptx_fast_periodic_exit entered
LACP: lacp_p(Gi1/0/1) timer stopped
LACP: Gi1/0/1 lacp_action_ptx_slow_periodic entered
LACP: timer lacp_p_s(Gi1/0/1) started with interval 30000.

LACP: recordPDU Gi1/0/1 Partner in sync and aggregating    <-- peer is in sync
LACP: Gi1/0/1 Partners oper state is hex 3D    <-- operational state update

LACP: timer lacp_c_l(Gi1/0/1) started with interval 90000.
LACP: Gi1/0/1 LAG_PARTNER_UP.
LACP: Gi1/0/1 LAG unchanged
    lacp_mux Gi1/0/1 - mux: during state COLLECTING_DISTRIBUTING, got event 5(in_sync) (ignored)
lacp_handle_standby_port_internal called, depth = 1
LACP: lacp_handle_standby_port_internal: No Standby port found for LAG 1
lacp_handle_standby_port_internal called, depth = 1
LACP: lacp_handle_standby_port_internal: No Standby port found for LAG 1
lacp_handle_standby_port_internal called, depth = 1
LACP: lacp_handle_standby_port_internal: No Standby port found for LAG 1
```

```
LACP: lacp_t(Gi1/0/1) timer stopped
LACP: lacp_t(Gi1/0/1) expired

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/2, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/3, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/4, changed state to up

%LINK-3-UPDOWN: Interface Port-channel1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1, changed state to up
```

If you focus on the two most important lines of the LACP debugs, there are a few concepts worth to define some LACP PDUs concepts.

<#root>

```
LACP:

 Act

: tlv:1, tlv-len:20,

key:0x1

, p-pri:0x8000, p:0x102,

p-state:0x3D

, s-pri:0x8000,

s-mac:f04a.0205.d600


LACP:

Part

: tlv:2, tlv-len:20,

key:0x1

, p-pri:0x8000, p:0x102,

p-state:0x3D

, s-pri:0x8000,

s-mac:f04a.0206.1900
```

Concept	Description
Act	Stands for actor (you)
Part	Stands for partner (your neighbor/peer)
key	It is the number of the port channel configured.

p-state	Stands for port state and it is the most important concept. It is built with 8 bits (LACP flags). Check the **Background Information** section for further information
s-mac	It is the system mac address used by LACP

Note: Values seen on debugs are hexadecimal, to properly read the values they must be translated to decimal or binary systems.

Verify PAgP Operation

This section describes how to verify the correct state and operation of the PAgP protocol.

Basic Checks

Check the PAgP outputs with these commands:

<#root>

```
show pagp <channel-group number> neighbor
```

```
show pagp <channel-group number> counters
```

```
show interfaces <interface ID> accounting
```

Check the details of the PAgP neighbor, such as the operational mode, partner system ID, hostname, and priority.

<#root>

```
switch#
show pagp 1 neighbor

Flags:  S - Device is sending Slow hello.  C - Device is in Consistent state.
        A - Device is in Auto mode.        P - Device learns on physical port.

Channel group 1 neighbors
            Partner
Partner

         Partner                   Partner Group
Port        Name
Device ID
```

```
      Port        Age  Flags    Cap.
Gi1/0/1     switch

f04a.0205.d600

   Gi1/0/1     16s  SC       10001

<-- Dev ID: Neighbor MAC Address

Gi1/0/2     switch

f04a.0205.d600

   Gi1/0/2     19s  SC       10001

<-- Dev ID: Neighbor MAC Address

Gi1/0/3     switch

f04a.0205.d600

   Gi1/0/3     17s  SC       10001

<-- Dev ID: Neighbor MAC Address

Gi1/0/4     switch

f04a.0205.d600

   Gi1/0/4     15s  SC       10001

<-- Dev ID: Neighbor MAC Address
```

Validate the output details of the PAgP packets sent and received by each interface. If corrupt PAgP packets are detected, the Pkts Err counter increases.

<#root>

```
switch#

show pagp 1 counters


            Information         Flush        PAgP
Port        Sent    Recv     Sent    Recv    Err Pkts
-----------------------------------------------------
Channel group: 1
Gi1/0/1

29      17

       0       0

 0


Gi1/0/2

 28      17

      0       0

 0
```

```
Gi1/0/3
  28      16
      0       0
 0

Gi1/0/4
 29      16
       0       0
0
```

There is also an option to check the interface accounting for PAgP.

<#root>

```
switch#
show int gi1/0/1 accounting

GigabitEthernet1/0/1
                Protocol    Pkts In   Chars In   Pkts Out  Chars Out
                   Other          0          0      10677     640620
                    PAgP        879      78231        891      79299
           Spanning Tree        240      12720         85       5100
                     CDP       2179     936495       2180     937020
                     DTP       3545     170160       3545     212700
                    LACP       3102     384648       3127     387748
```

Debugs

If you notice PAgP negotiation issues, **enable** PAgP debugs to analyze why.

<#root>

```
switch#
debug pagp event

Port Aggregation Protocol events debugging is on
switch#
debug pagp packet

Port Aggregation Protocol packet debugging is on
switch#
debug pagp fsm
```

```
Port Aggregation Protocol fsm debugging is on
switch#
```

debug pagp misc

```
Port Aggregation Protocol miscellaneous debugging is on
```

If needed, **enable** debug condition to a specific interface and filter the output.

<#root>

```
switch#
```

debug condition interface gigabitEthernet 1/0/1

Note: PAgP debugs are platform agnostic.

Validate debugs and filters are set up.

<#root>

```
switch#
```

show debugging

```
Packet Infra debugs:

Ip Address                                                Port
-------------------------------------------------------|----------

PAGP:
  Port Aggregation Protocol
```

miscellaneous

```
 debugging is
```

on

```
  Port Aggregation Protocol
```

packet

```
 debugging is
```

on

```
  Port Aggregation Protocol
```

fsm

```
 debugging is
```

on

```
  Port Aggregation Protocol
events
 debugging is
on


Condition 1: interface Gi1/0/1 (1 flags triggered)
        Flags: Gi1/0/1
```

Analyze the PAgP debugs. The debug output shows the last PAgP frames before the port-channel interface comes up:

<#root>

```
PAgP: Receive information packet via Gi1/0/1, packet size: 89
flags: 5, my device ID: f04a.0205.d600, learn-cap: 2, port-priority: 128, sent-port-ifindex: 9, group-ca
your device ID: f04a.0206.1900, learn-cap: 2, port-priority: 128, sent-port-ifindex: 9, group-cap: 10001


partner count: 1, num-tlvs: 2
device name TLV: switch
port name TLV: Gi1/0/1
PAgP: Gi1/0/1 PAgP packet received, processing    <-- Processing ingress PAgP frame
PAgP: Gi1/0/1 proved to be bidirectional    <--

PAgP: Gi1/0/1 action_b0 is entered
PAgP: Gi1/0/1 Input = Transmission State, V12 Old State = U5  New State = U5
PAgP: Gi1/0/1 action_a6 is entered
PAgP: Gi1/0/1 action_b9 is entered
PAgP: set hello interval from 1000 to 30000 for port Gi1/0/1    <--

PAgP: Gi1/0/1 Input = Transmission State, V10 Old State = U5  New State = U6
PAgP: set partner 0 interval from 3500 to 105000 for port Gi1/0/1
PAgP: Gi1/0/1 Setting hello flag
PAgP: timer pagp_p(Gi1/0/1) started with interval 105000.
PAgP: pagp_i(Gi1/0/1) timer stopped
PAgP: Gi1/0/1 Input = Port State, E5 Old State = S7  New State = S7
PAgP: pagp_h(Gi1/0/1) expired


PAgP: Send information packet via Gi1/0/1, packet size: 89
flags: 5, my device ID: f04a.0206.1900, learn-cap: 2, port-priority: 128, sent-port-ifindex: 9, group-ca
your device ID: f04a.0205.d600, learn-cap: 2, port-priority: 128, sent-port-ifindex: 9, group-cap: 10001


partner count: 1, num-tlvs: 2
device name TLV: switch
port name TLV: Gi1/0/1
PAgP: 89 bytes out Gi1/0/1
PAgP: Gi1/0/1 Transmitting information packet

PAgP: timer pagp_h(Gi1/0/1) started with interval 30000    <--
%LINK-3-UPDOWN: Interface Port-channel1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel1, changed state to up
```

Verify Etherchannel Programming

This section describes how to verify the software and hardware settings for EtherChannel.

Verify Software

Validate the software entries.

<#root>

```
show run interface <interface ID>
```

```
show etherchannel <channel-group number> summary
```

Check the EtherChannel configuration.

<#root>

```
switch#
show run int gigabitEthernet 1/0/1

<output omitted>
interface GigabitEthernet1/0/1
 channel-group 1 mode active
end

switch#
show run int gigabitEthernet 1/0/2

<output omitted>
interface GigabitEthernet1/0/2
 channel-group 1 mode active
end

switch#
show run int gigabitEthernet 1/0/3

<output omitted>
interface GigabitEthernet1/0/3
 channel-group 1 mode active
end

switch#
show run int gigabitEthernet 1/0/4

<output omitted>
interface GigabitEthernet1/0/4
 channel-group 1 mode active
end

switch#
```

```
show run int port-channel 1
```

```
<output omitted>
interface Port-channel1
end
```

Validate all port members are bundled in the port channel.

<#root>

```
switch#
```

```
show etherchannel 1 summary
```

```
<output omitted>
Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)         LACP        Gi1/0/1(P)      Gi1/0/2(P)
                                   Gi1/0/3(P)      Gi1/0/4(P)
```

Verify Hardware

Validate software entries at the hardware level:

<#root>

```
show platform software interface switch <switch number or role> r0 br
```

```
show platform software fed switch <switch number or role> etherchannel <channel-group number> group-mask
```

```
show platform software fed switch <switch number or role> ifm mappings etherchannel
```

```
show platform software fed switch <switch number or role> ifm if-id <if ID>
```

Check the ID of the port channel and bundled interfaces.

<#root>

```
switch#
```

```
show platform software interface switch active r0 br
```

```
Forwarding Manager Interfaces Information
```

```
Name
```

```
ID
```

```
               QFP ID
-------------------------------------------------------------
<output omitted>
GigabitEthernet1/0/1

9

                 0
GigabitEthernet1/0/2

10

                0
GigabitEthernet1/0/3

11

                0
GigabitEthernet1/0/4

12

                0
<output omitted>
Port-channel1

76

                0
```

Focus on the **IF ID** section and ensure the value (hexadecimal number) is equivalent to the **ID** (decimal number) observed in the previous command.

<#root>

```
switch#

show platform software fed switch active etherchannel 1 group-mask


Group Mask Info

Aggport IIF Id: 000000000000004c    <-- IfId Hex 0x4c = 76 decimal

Active Port: : 4

Member Ports
If Name

If Id

             local  Group Mask
----------------------------------------------------------------------
GigabitEthernet1/0/4

000000000000000c

  true   7777777777777777

<-- IfId Hex 0xc = 12 decimal


GigabitEthernet1/0/3
```

```
000000000000000b

  true   bbbbbbbbbbbbbbbb

<-- IfId Hex 0xb = 11 decimal


GigabitEthernet1/0/2

000000000000000a

  true   ddddddddddddddddd

<-- IfId Hex 0xa = 10 decimal


GigabitEthernet1/0/1

0000000000000009

  true   eeeeeeeeeeeeeeee

<-- IfId Hex 0x9 = 10 decimal
```

Obtain the **IF ID** of the port channel with the next command. The value must match the one from the earlier command.

<#root>

```
Switch#

show platform software fed switch active ifm mappings etherchannel


Mappings Table

Chan Interface IF_ID
-----------------------------------------------------
1 Port-channel1

0x0000004c
```

Use the **IF ID** for the next command. The information shown must match with the outputs collected earlier.

<#root>

```
switch#

show platform software fed switch active ifm if-id 0x0000004c

Interface IF_ID         : 0x000000000000004c
Interface Name          : Port-channel1

Interface Block Pointer : 0x7f0178ca1a28
Interface Block State   : READY
Interface State         : Enabled
Interface Status        : ADD, UPD
Interface Ref-Cnt       : 8
```

```
Interface Type          : ETHERCHANNEL
        Port Type         : SWITCH PORT
        Channel Number    : 1

        SNMP IF Index     : 78
        Port Handle       : 0xdd000068
        # Of Active Ports : 4
        Base GPN          : 1536

                Index[2]  : 000000000000000c
                Index[3]  : 000000000000000b
                Index[4]  : 000000000000000a
                Index[5]  : 0000000000000009


        Port Information
        Handle ............ [0xdd000068]

        Type ............... [L2-Ethchannel]

        Identifier ........ [0x4c]
        Unit ............... [1]

        DI ................ [0x7f0178c058a8]
        Port Logical Subblock
                L3IF_LE handle .... [0x0]
                Num physical port . [4]
                GPN Base .......... [1536]
                Physical Port[2] .. [0x7b000027]
                Physical Port[3] .. [0x1f000026]
                Physical Port[4] .. [0xc000025]
                Physical Port[5] .. [0xb7000024]
                Num physical port on asic [0] is  [0]
                DiBcam handle on asic [0].... [0x0]
                Num physical port on asic [1] is  [4]
                DiBcam handle on asic [1].... [0x7f0178c850a8]
                SubIf count ....... [0]
        Port L2 Subblock
                Enabled ............ [No]
                Allow dot1q ......... [No]
                Allow native ........ [No]
                Default VLAN ........ [0]
                Allow priority tag ... [No]
                Allow unknown unicast  [No]
                Allow unknown multicast[No]
                Allow unknown broadcast[No]
                Allow unknown multicast[Enabled]
                Allow unknown unicast  [Enabled]
                Protected ........... [No]
                IPv4 ARP snoop ....... [No]
                IPv6 ARP snoop ....... [No]
                Jumbo MTU ........... [0]
                Learning Mode ........ [0]
                Vepa ................ [Disabled]
                App Hosting........... [Disabled]
        Port QoS Subblock
                Trust Type ..................... [0x7]
                Default Value ................. [0]
                Ingress Table Map ............. [0x0]
                Egress Table Map .............. [0x0]
                Queue Map ..................... [0x0]
        Port Netflow Subblock
        Port Policy Subblock
```

```
        List of Ingress Policies attached to an interface
        List of Egress Policies attached to an interface
        Port CTS Subblock
                Disable SGACL ..................... [0x0]
                Trust ............................ [0x0]
                Propagate ........................ [0x0]
                Port SGT ......................... [0xffff]
Ref Count : 8 (feature Ref Counts + 1)
IFM Feature Ref Counts
        FID : 97 (AAL_FEATURE_L2_MULTICAST_IGMP), Ref Count : 1
        FID : 119 ((null)), Ref Count : 1
        FID : 84 (AAL_FEATURE_L2_MATM), Ref Count : 1
No Sub Blocks Present
```

Platform Tools

This table shows what tools and features are available in order to help understand when to use them:

Tool	Level	When to use it
EPC	Hardware and software	Use it to validate LACP frames landed at the physical interface or to validate they reach out to the CPU.
Platform Forward	Hardware	If you confirmed LACP frames landed on the switch, use this tool to know the internal forwarding decision of the switch.
PSV	Hardware	If you confirmed LACP frames landed on the switch, use this tool to know the internal forwarding decision of the switch.
CoPP	Hardware	If the packet was forwarded to the CPU from a hardware perspective, however, was not seen at the software (CPU) level. It is very likely this feature dropped the LACP frame along the path between the hardware and the CPU.
FED CPU packet capture	Software	Use it to validate that the LACP frame was punted to the CPU through the right queue, it also validates if the CPU sends LACP frames back to the hardware.

> **Note**: Only LACP protocol is analyzed with the use of these tools, however, they can also be used to analyze PAgP frames.

Embedded Packet Capture (EPC)

The commands to set up the Wireshark (EPC) and capture ingress/egress LACP PDUs.

<#root>

```
monitor capture <capture name> [control-plane|interface <interface ID>] BOTH


monitor capture <capture name> match mac [any|host <source MAC address>|<source MAC address>][any|host <


monitor capture <capture name> file location flash:<name>.pcap


show monitor capture <capture name> parameter


show monitor capture <capture name>


monitor capture <capture name> start


monitor capture <capture name> stop


show monitor capture file flash:<name>.pcap [detailed]
```

Note: Commands are entered under privilege mode.

Set up the Wireshark capture.

Tip: If you want to focus on a specific bundled interface and/or specific source MAC address tune the **interface** and **match mac** keywords.

<#root>

```
monitor capture CAP interface GigabitEthernet1/0/1 BOTH


monitor capture CAP interface GigabitEthernet1/0/2 BOTH


monitor capture CAP interface GigabitEthernet1/0/3 BOTH


monitor capture CAP interface GigabitEthernet1/0/4 BOTH


monitor capture CAP match mac any host 0180.c200.0002


show monitor capture CAP file location flash:CAP.pcap
```

Note: Destination MAC address **0180.c200.0002** defined on the capture helps you to filter LACP frames.

Verify the Wireshark was configured properly:

<#root>

```
switch#
```

```**show mon cap CAP parameter**

monitor capture CAP interface GigabitEthernet1/0/1 BOTH
monitor capture CAP interface GigabitEthernet1/0/2 BOTH
monitor capture CAP interface GigabitEthernet1/0/3 BOTH
monitor capture CAP interface GigabitEthernet1/0/4 BOTH
monitor capture CAP match mac any host 0180.c200.0002
monitor capture CAP file location flash:LACP.pcap
```

```
switch#
```

```**show mon cap CAP**

Status Information for Capture CAP
  Target Type:
 Interface: GigabitEthernet1/0/1, Direction: BOTH
 Interface: GigabitEthernet1/0/2, Direction: BOTH
 Interface: GigabitEthernet1/0/3, Direction: BOTH
 Interface: GigabitEthernet1/0/4, Direction: BOTH
   Status : Inactive
  Filter Details:
   MAC
     Source MAC: 0000.0000.0000 mask:ffff.ffff.ffff
     Destination MAC: 0180.c200.0002 mask:0000.0000.0000
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:CAP.pcap
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Start the capture:

<#root>

```
switch#
```

```**mon cap CAP start**

Started capture point : CAP
```

Stop it after (at least) 30 seconds if you do not use LACP rate fast timer:

<#root>

```
switch#
```

```**mon cap CAP stop**```

```
Capture statistics collected at software:
        Capture duration - 58 seconds
        Packets received - 16
        Packets dropped - 0
        Packets oversized - 0

Bytes dropped in asic - 0

Stopped capture point : CAP
```

Frames captured:

<#root>

```
switch#
```

```**show mon cap file flash:CAP.pcap**```

```
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

    1   0.000000 f0:4a:02:06:19:04 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 261 K:
    2   2.563406 f0:4a:02:05:d6:01 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 258 K:
    3   3.325148 f0:4a:02:05:d6:04 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 261 K:
    4   5.105978 f0:4a:02:06:19:01 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 258 K:
    5   6.621438 f0:4a:02:06:19:02 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 259 K:
    6   8.797498 f0:4a:02:05:d6:03 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 260 K:
    7  13.438561 f0:4a:02:05:d6:02 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 259 K:
    8  16.658497 f0:4a:02:06:19:03 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 260 K:
    9  28.862344 f0:4a:02:06:19:04 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 261 K:
   10  29.013031 f0:4a:02:05:d6:01 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 258 K:
   11  30.756138 f0:4a:02:05:d6:04 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 261 K:
   12  33.290542 f0:4a:02:06:19:01 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 258 K:
   13  36.387119 f0:4a:02:06:19:02 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 259 K:
   14  37.598788 f0:4a:02:05:d6:03 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 260 K:
   15  40.659931 f0:4a:02:05:d6:02 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:05:d6:00 P: 259 K:
   16  45.242014 f0:4a:02:06:19:03 b^F^R 01:80:c2:00:00:02 LACP 124 v1 ACTOR f0:4a:02:06:19:00 P: 260 K:
```

If you need to check the LACP field from a specific frame use the **detailed** keyword.

<#root>

```
switch#
```

```**show mon cap file flash:CAP.pcap detailed**```

```
Starting the packet display ........ Press Ctrl + Shift + 6 to exit
```
```

Frame 1: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)

```
 on interface 0
    Interface id: 0 (/tmp/epc_ws/wif_to_ts_pipe)
        Interface name: /tmp/epc_ws/wif_to_ts_pipe
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 28, 2023 15:48:14.985430000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1680018494.985430000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 124 bytes (992 bits)
    Capture Length: 124 bytes (992 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:slow:lacp]
```

Ethernet II, Src: f0:4a:02:06:19:04 (f0:4a:02:06:19:04), Dst: 01:80:c2:00:00:02 (01:80:c2:00:00:02)

```
    Destination: 01:80:c2:00:00:02 (01:80:c2:00:00:02)
        Address: 01:80:c2:00:00:02 (01:80:c2:00:00:02)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
    Source: f0:4a:02:06:19:04 (f0:4a:02:06:19:04)
        Address: f0:4a:02:06:19:04 (f0:4a:02:06:19:04)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: Slow Protocols (0x8809)
```

Slow Protocols
    Slow Protocols subtype: LACP (0x01)
Link Aggregation Control Protocol

```
    LACP Version: 0x01
    TLV Type: Actor Information (0x01)
    TLV Length: 0x14
    Actor System Priority: 32768
    Actor System ID: f0:4a:02:06:19:00 (f0:4a:02:06:19:00)
    Actor Key: 1
    Actor Port Priority: 32768
    Actor Port: 261
    Actor State: 0x3d, LACP Activity, Aggregation, Synchronization, Collecting, Distributing
        .... ...1 = LACP Activity: Active
        .... ..0. = LACP Timeout: Long Timeout
        .... .1.. = Aggregation: Aggregatable
        .... 1... = Synchronization: In Sync
        ...1 .... = Collecting: Enabled
        ..1. .... = Distributing: Enabled
        .0.. .... = Defaulted: No
        0... .... = Expired: No
    [Actor State Flags: **DCSG*A]
    Reserved: 000000
    TLV Type: Partner Information (0x02)
    TLV Length: 0x14
    Partner System Priority: 32768
    Partner System: f0:4a:02:05:d6:00 (f0:4a:02:05:d6:00)
    Partner Key: 1
    Partner Port Priority: 32768
    Partner Port: 261
    Partner State: 0x3d, LACP Activity, Aggregation, Synchronization, Collecting, Distributing
```

```
        .... ...1 = LACP Activity: Active
        .... ..0. = LACP Timeout: Long Timeout
        .... .1.. = Aggregation: Aggregatable
        .... 1... = Synchronization: In Sync
        ...1 .... = Collecting: Enabled
        ..1. .... = Distributing: Enabled
        .0.. .... = Defaulted: No
        0... .... = Expired: No
    [Partner State Flags: **DCSG*A]
    Reserved: 000000
    TLV Type: Collector Information (0x03)
    TLV Length: 0x10
    Collector Max Delay: 32768
    Reserved: 000000000000000000000000
    TLV Type: Terminator (0x00)
    TLV Length: 0x00
    Pad: 000000000000000000000000000000000000000000000000...

Frame 2: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0
    Interface id: 0 (/tmp/epc_ws/wif_to_ts_pipe)
        Interface name: /tmp/epc_ws/wif_to_ts_pipe
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 28, 2023 15:48:17.548836000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1680018497.548836000 seconds
    [Time delta from previous captured frame: 2.563406000 seconds]
    [Time delta from previous displayed frame: 2.563406000 seconds]
    [Time since reference or first frame: 2.563406000 seconds]
```

Note: Wireshark output format can differ on 9200 devices and not be readable from the switch. Export the capture and read it from your PC if that is the case.

Platform Forward

In order to debug forwarding information and to trace the packet path in the hardware forwarding plane, use the show platform hardware fed switch <switch number or role> forward interface command. This command simulates a user-defined packet and retrieves the forwarding information from the hardware forwarding plane. A packet is generated on the ingress port based on the packet parameters that you have specified in this command. You can also provide a complete packet from the captured packets stored in a PCAP file.

This topic elaborates only on the interface forwarding-specific options, that is, the options available with the show platform hardware fed switch {switch_num|active|standby}forward interface command.

<#root>

```
show platform hardware fed switch <switch number or role> forward interface <interface ID> <source mac a
show platform hardware fed switch <switch number or role> forward interface <interface ID> pcap <pcap fi
show platform hardware fed switch <switch number or role> forward interface <interface ID> vlan <VLAN ID
```

Define the Platform Forward capture. In this case, CAP.pcap frame **1** is analyzed.

<#root>

```
switch#
show platform hardware fed switch active forward interface gigabitEthernet 1/0/1 pcap flash:CAP.pcap num
show forward is running in the background. After completion, syslog will be generated.
```

Once Platform Forward capture is done, the next Syslog messages are shown.

<#root>

```
switch#
show logging

<output omitted>
*Mar 28 16:47:57.289: %SHFWD-6-PACKET_TRACE_DONE: Switch 1 R0/0: fed: Packet Trace Complete: Execute (sh
*Mar 28 16:47:57.289: %SHFWD-6-PACKET_TRACE_FLOW_ID: Switch 1 R0/0: fed: Packet Trace Flow id is 1009909
```

Analyze the Platform Forward capture. The **Egress** section tells you what the internal forwarding decision was. LACP and PAgP frames are expected to be punted to the CPU.

<#root>

```
switch#
show platform hardware fed switch active forward last summary

Input Packet Details:
###[ Ethernet ]###
  dst       = 01:80:c2:00:00:02
  src.      = f0:4a:02:06:19:04
  type      = 0x8809   <-- slow protocols (LACP) defined by IANA

###[ Raw ]###
     load      = '01 01 01 14 80 00 F0 4A 02 06 19 00 00 01 80 00 01 05 3D 00 00 00 02 14 80 00 F0 4A 02
Ingress:
   Port                      :
   Global Port Number        : 1536
   Local Port Number         : 0
   Asic Port Number          : 0
   Asic Instance             : 1
   Vlan                      : 1
   Mapped Vlan ID            : 4
   STP Instance              : 2
   BlockForward              : 0
   BlockLearn                : 0
   L3 Interface              : 37
       IPv4 Routing          : enabled
       IPv6 Routing          : enabled
       Vrf Id                : 0
   Adjacency:
       Station Index         : 107    [SI_CPUQ_L2_CONTROL]
       Destination Index     : 21106
       Rewrite Index         : 1
       Replication Bit Map   : 0x20   ['coreCpu']
```

```
   Decision:
       Destination Index     : 21106  [DI_CPUQ_L2_CONTROL]
       Rewrite Index         : 1      [RI_CPU]
       Dest Mod Index        : 0      [IGR_FIXED_DMI_NULL_VALUE]
       CPU Map Index         : 0      [CMI_NULL]
       Forwarding Mode       : 0      [Bridging]
       Replication Bit Map   :        ['coreCpu']
       Winner                :        L2DESTMACVLAN LOOKUP
       Qos Label             : 65
       SGT                   : 0
       DGTID                 : 0

Egress:
   Possible Replication      :
       Port                  : CPU_Q_L2_CONTROL
   Output Port Data          :
     Port                    : CPU

       Asic Instance         : 0

       CPU Queue             :      1 [CPU_Q_L2_CONTROL]

       Unique RI             : 0
       Rewrite Type          : 0      [NULL]
       Mapped Rewrite Type   : 15     [CPU_ENCAP]

       Vlan                  : 1


      Mapped Vlan ID        : 4
*******************************************************************************
```

Packet State Vector (PSV)

PSV is similar to Platform Forward captures with the exception that PSV captures live ingress frames from the network that matches the trigger criteria.

> **Note**: PSV is only supported on C9500-32C, C9500-32QC, C9500-24Y4C, C9500-48Y4C, and C9606R platforms.

<#root>

```
debug platform hardware fed <switch number or role> capture trigger interface <interface ID> ingress


debug platform hardware fed <switch number or role> capture trigger layer2 <source MAC address> <destina


show platform hardware fed <switch number or role> capture trigger


show platform hardware fed <switch number or role> capture status


show platform hardware fed <switch number or role> capture summary
```

Two C9500-48Y4C connected to each other are used for the next port channel and PSV capture.

<#root>

```
switch#
```

```**show etherchannel 1 summary**```

```
<output omitted>
Group Port-channel Protocol Ports
------+-------------+-----------+-----------------------------------------------
1 Po1(SU) LACP
```

```**Twe1/0/1(P)**```

```
 Twe1/0/2(P)
```

Set up the trigger criteria. Use the **layer2** keyword to match with the specific source MAC address and the LACP MAC address as the destination.

<#root>

```
switch#debug platform hardware fed active capture trigger interface twentyFiveGigE1/0/1 ingress
switch#debug platform hardware fed active capture trigger layer2
```

```**0000.0000.0000 0180.c200.0002 <-- match source MAC: any, match destination MAC: LACP MAC address**```

```
Capture trigger set successful.
```

Note: MAC address **0000.0000.0000** defined on the PSV capture means **match any**.

Validate trigger criteria were setup.

<#root>

```
switch#
```

```**show platform hardware fed active capture trigger**```

```
Trigger Set:
Ingress Interface: TwentyFiveGigE1/0/1
Dest Mac: 0180.c200.0002
```

Once PST has been triggered, the status is shown as **Completed**.

<#root>

```
switch#
show platform hardware fed active capture status

Asic: 0
Status: Completed
```

Analyze the PSV capture output with the next command. It is expected to see LACP and PAgP frames are punted to the CPU.

<#root>

```
switch#
show platform hardware fed active capture summary

Trigger: Ingress Interface:TwentyFiveGigE1/0/1 Dest Mac:0180.c200.0002
Input        Output       State     Reason
Tw1/0/1      cpuQ 1       PUNT
Bridged
```

Control Plane Policer (CoPP)

CoPP is basically a QoS policer applied to the pipe between the data plane (hardware) and the control plane (CPU) to avoid high CPU issues. CoPP can filter LACP and PAgP frames if these frames exceed the threshold established by the feature.

Validate if CoPP drops LACP packets.

<#root>

```
show platform hardware fed switch active qos queue stats internal cpu policer
```

The output of this command, **L2 Control** queue has no drops:

<#root>

```
switch#
show platform hardware fed switch active qos queue stats internal cpu policer

                          CPU Queue Statistics
============================================================================================
                                                (default)
```

```
(set)

     Queue        Queue

QId PlcIdx



Queue Name

                Enabled   Rate

Rate

      Drop(Bytes)  Drop(Frames)
-----------------------------------------------------------------------------------------------
0    11     DOT1X Auth                  Yes     1000      1000     0            0

1    1      L2 Control                  Yes     2000      2000     0            0          <-- L2 Contro

2    14     Forus traffic               Yes     4000      4000     0            0

<output omitted>

* NOTE: CPU queue policer rates are configured to the closest hardware supported value

                      CPU Queue Policer Statistics
=================================================================
Policer    Policer Accept   Policer Accept  Policer Drop  Policer Drop
  Index         Bytes          Frames        Bytes          Frames
-----------------------------------------------------------------
0          0                0               0             0

1          13328202         79853           0             0          <-- QId = 1 matches policer index (

2          0                0               0             0

<output omitted>

                  Second Level Policer Statistics
=================================================================

20         34149506         389054          0             0          <-- Policer index (level 2) no drop

21         76896            596             0             0

Policer Index Mapping and Settings
-----------------------------------------------------------------
level-2   :   level-1                     (default)   (set)
PlcIndex  :   PlcIndex                      rate      rate
-----------------------------------------------------------------

20        :   1  2  8                       13000     13000       <-- Policer index (level 1) = 1 match

21        :   0 4 7 9 10 11 12 13 14 15     6000      6000
=================================================================

               Second Level Policer Config
=================================================================
    level-1 level-2                           level-2
QId PlcIdx  PlcIdx  Queue Name                Enabled
-----------------------------------------------------------------
0    11      21      DOT1X Auth                 Yes

1    1       20      L2 Control                 Yes
```

```
2    14      21      Forus traffic              Yes

<output omitted>
```

It is not expected to overwhelm the **L2 Control** queue. Control plane packet capture is needed when the opposite is observed.

FED CPU Packet Capture

If you have ensured LACP packets were received at the interface level, EPC and ELAM/PSV confirmed LACP frames were punted to the CPU with no drops observed at the CoPP level, then use the FED CPU packet capture tool.

FED CPU packet capture tells you why a packet was punted from hardware to CPU, it also tells you what CPU queue the packet was sent to. FED CPU packet capture can also capture packets generated by the CPU injected into hardware.

<#root>

```
debug platform software fed sw active punt packet-capture set-filter <filter>


debug platform software fed switch active punt packet-capture start


debug platform software fed switch active punt packet-capture stop


show platform software fed switch active punt packet-capture status


show platform software fed switch active punt packet-capture brief


debug platform software fed sw active inject packet-capture set-filter <filter>


debug platform software fed switch active inject packet-capture start


debug platform software fed switch active inject packet-capture stop


show platform software fed switch active inject packet-capture status


show platform software fed switch active inject packet-capture brief
```

Punt

Define the packet capture to filter only LACP packets.

<#root>

```
switch#
debug platform software fed sw active punt packet-capture set-filter "eth.dst==0180.c200.0002"

Filter setup successful. Captured packets will be cleared
```

Start the capture.

<#root>

```
switch#
debug platform software fed sw active punt packet-capture start

Punt packet capturing started.
```

Stop it after (at least) 30 seconds if you do not use LACP rate fast timer.

<#root>

```
switch#
debug platform software fed switch active punt packet-capture stop

Punt packet capturing stopped.
Captured 11 packet(s)
```

Check the FED CPU packet capture status.

<#root>

```
switch#
show platform software fed switch active punt packet-capture status

Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 11 packets.
 Capture capacity : 4096 packets
Capture filter : "eth.dst==0180.c200.0002"
```

Analyze the FED CPU packet capture output.

<#root>

```
switch#
show platform software fed switch active punt packet-capture brief
Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 11 packets
. Capture capacity : 4096 packets
Capture filter : "eth.dst==0180.c200.0002"
------ Punt Packet Number: 1, Timestamp: 2023/03/31 00:27:54.141 ------
 interface :
physical: GigabitEthernet1/0/2[if-id: 0x0000000a]
, pal: GigabitEthernet1/0/2 [if-id: 0x0000000a]
<-- interface that punted the frame
 metadata  :
cause: 96 [Layer2 control protocols],
 sub-cause: 0,
q-no: 1
, linktype: MCP_LINK_TYPE_LAYER2 [10]
<-- LACP frame was punted due to L2 ctrl protocol to queue 1 (L2 control)
 ether hdr :
dest mac: 0180.c200.0002, src mac: f04a.0205.d602    <-- source and destination MAC addresses
 ether hdr : ethertype: 0x8809
------ Punt Packet Number: 2, Timestamp: 2023/03/31 00:27:58.436 ------
 interface :
physical: GigabitEthernet1/0/4[if-id: 0x0000000c]
, pal: GigabitEthernet1/0/4 [if-id: 0x0000000c]
 metadata  :
cause: 96 [Layer2 control protocols]
, sub-cause: 0,
q-no: 1
, linktype: MCP_LINK_TYPE_LAYER2 [10]
 ether hdr : dest mac: 0180.c200.0002,
src mac: f04a.0205.d604
 ether hdr : ethertype: 0x8809
------ Punt Packet Number: 3, Timestamp: 2023/03/31 00:28:00.758 ------
 interface :
physical: GigabitEthernet1/0/1[if-id: 0x00000009]
```

```
, pal: GigabitEthernet1/0/1 [if-id: 0x00000009]
 metadata  :
```

cause: 96 [Layer2 control protocols]

```
, sub-cause: 0,
```

q-no: 1

```
, linktype: MCP_LINK_TYPE_LAYER2 [10]
 ether hdr : dest mac: 0180.c200.0002,
```

src mac: f04a.0205.d601

```
 ether hdr : ethertype: 0x8809

------ Punt Packet Number: 4, Timestamp: 2023/03/31 00:28:11.888 ------
 interface :
```

physical: GigabitEthernet1/0/3[if-id: 0x0000000b]

```
, pal: GigabitEthernet1/0/3 [if-id: 0x0000000b]
 metadata  :
```

cause: 96 [Layer2 control protocols]

```
, sub-cause: 0,
```

q-no: 1

```
, linktype: MCP_LINK_TYPE_LAYER2 [10]
 ether hdr : dest mac: 0180.c200.0002,
```

src mac: f04a.0205.d603

```
 ether hdr : ethertype: 0x8809
```

Inject

Define the packet capture to filter only LACP packets.

<#root>

```
switch#
```

debug platform software fed sw active inject packet-capture set-filter "eth.dst==0180.c200.0002"

```
Filter setup successful. Captured packets will be cleared
```

Start the capture.

<#root>

```
switch#
```

debug platform software fed sw active inject packet-capture start

```
Punt packet capturing started.
```

Stop it after (at least) 30 seconds if you do not use LACP rate fast timer.

<#root>

```
switch#
debug platform software fed switch active inject packet-capture stop

Inject packet capturing stopped.
Captured 12 packet(s)
```

Check the FED CPU packet capture status.

<#root>

```
switch#
show platform software fed sw active inject packet-capture status

Inject packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 12 packets.
 Capture capacity : 4096 packets
Capture filter : "eth.dst==0180.c200.0002"
```

Analyze the FED CPU packet capture output.

<#root>

```
switch#
show platform software fed sw active inject packet-capture brief

Inject packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 12
 packets. Capture capacity : 4096 packets
Capture filter : "eth.dst==0180.c200.0002"

------ Inject Packet Number: 1, Timestamp: 2023/03/31 19:59:26.507 ------
 interface :
pal: GigabitEthernet1/0/2 [if-id: 0x0000000a]    <-- interface that LACP frame is destined to
 metadata  :
cause: 1 [L2 control/legacy]
, sub-cause: 0,
```

```
q-no: 7

, linktype: MCP_LINK_TYPE_LAYER2 [10]

    <-- cause L2 ctrl, queue=7 (high priority)

 ether hdr :

dest mac: 0180.c200.0002, src mac: f04a.0206.1902    <-- source and destination MAC addresses

 ether hdr : ethertype: 0x8809

------ Inject Packet Number: 2, Timestamp: 2023/03/31 19:59:28.538 ------
 interface :

pal: GigabitEthernet1/0/3 [if-id: 0x0000000b]

 metadata  :

cause: 1 [L2 control/legacy]

, sub-cause: 0,

q-no: 7

, linktype: MCP_LINK_TYPE_LAYER2 [10]
 ether hdr :

dest mac: 0180.c200.0002, src mac: f04a.0206.1903

 ether hdr : ethertype: 0x8809

------ Inject Packet Number: 3, Timestamp: 2023/03/31 19:59:30.050 ------
 interface :

pal: GigabitEthernet1/0/1 [if-id: 0x00000009]

 metadata  :

cause: 1 [L2 control/legacy]

, sub-cause: 0,

q-no: 7

, linktype: MCP_LINK_TYPE_LAYER2 [10]
 ether hdr :

dest mac: 0180.c200.0002, src mac: f04a.0206.1901

 ether hdr : ethertype: 0x8809

------ Inject Packet Number: 4, Timestamp: 2023/03/31 19:59:33.467 ------
 interface : pal:

GigabitEthernet1/0/4 [if-id: 0x0000000c]

 metadata  :

cause: 1 [L2 control/legacy]

, sub-cause: 0,

q-no: 7

, linktype: MCP_LINK_TYPE_LAYER2 [10]
 ether hdr :

 dest mac: 0180.c200.0002, src mac: f04a.0206.1904
```

```
ether hdr : ethertype: 0x8809
```

Related Information

- IEEE 802 Numbers
- IEEE - Link Aggregation Control Protocol
- Layer 2 Configuration Guide, Cisco IOS XE Amsterdam 17.3.x (Catalyst 9200 Switches) - Chapter: Configuring EtherChannels
- Layer 2 Configuration Guide, Cisco IOS XE Cupertino 17.7.x (Catalyst 9300 Switches) - Chapter: Configuring EtherChannels
- Layer 2 Configuration Guide, Cisco IOS XE Amsterdam 17.3.x (Catalyst 9400 Switches) - Chapter: Configuring EtherChannels
- Layer 2 Configuration Guide, Cisco IOS XE Cupertino 17.9.x (Catalyst 9500 Switches) - Chapter: Configuring EtherChannels
- Layer 2 Configuration Guide, Cisco IOS XE Cupertino 17.9.x (Catalyst 9600 Switches) - Chapter: Configuring EtherChannels
- Chapter: Interface and Hardware Commands - show platform hardware fed switch forward interface
- Configure FED CPU Packet Capture on Catalyst 9000 Switches
- Technical Support & Documentation - Cisco Systems